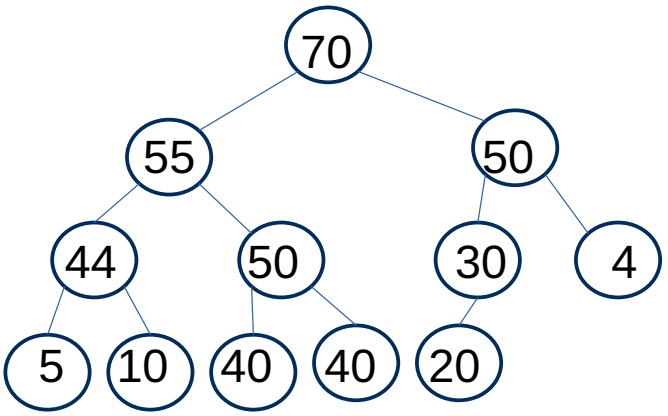
Descrição e exemplos com Max-Heap

Dado um heap binário do tipo Max-Heap

- Inicialmente VAZIO
- Vamos inserir os elementos:
  40 10 30 70 50 20 4 5 44 40 55 50

Após todas as insersões o Max-Heap fica:



Operações decreaseMax no Max-Heap

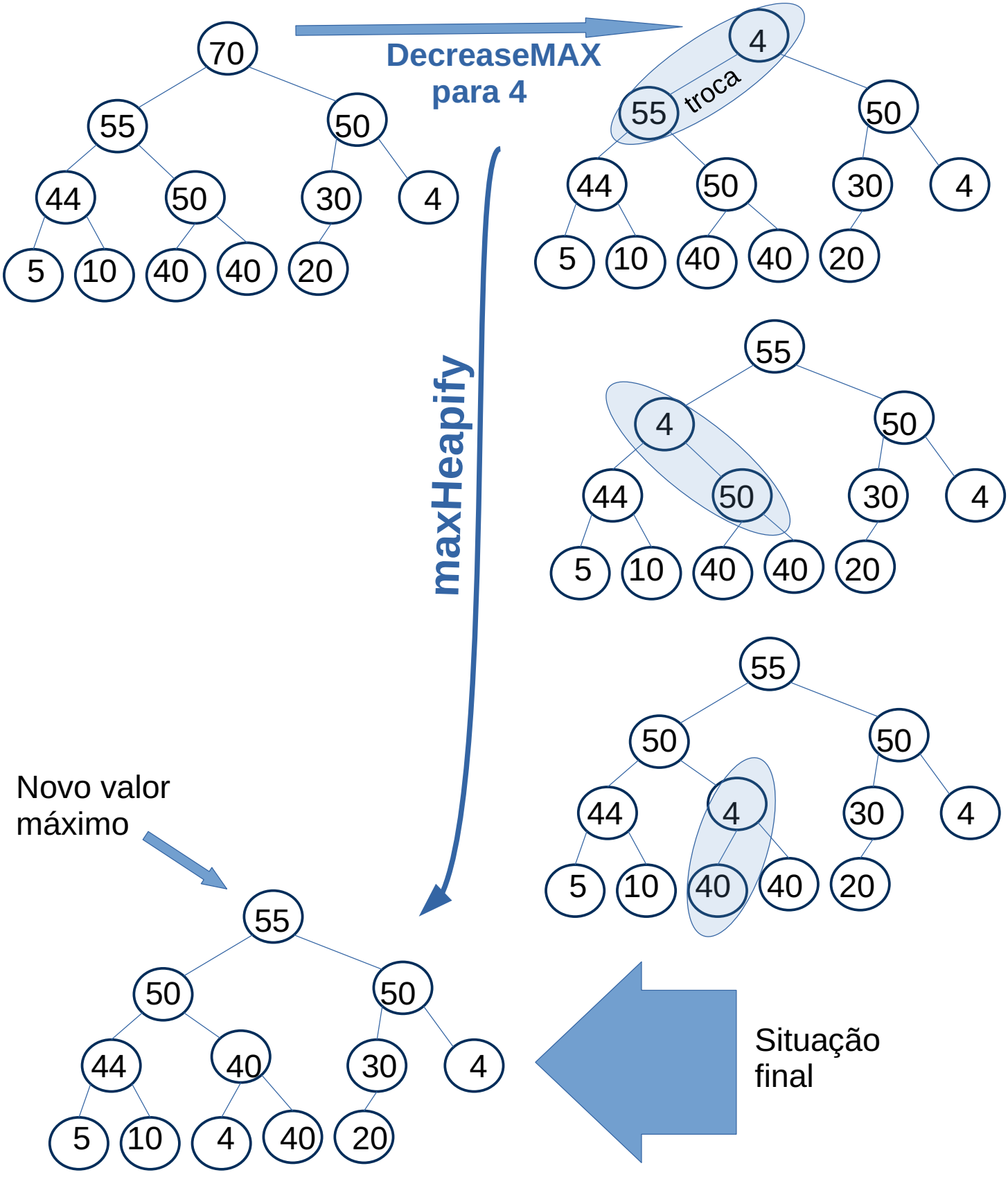
Definiremos a operação decreaseMax no Max-Heap

Dado um Max-Heap h com valor máximo max
(i.e. o valor na raiz do heap)

- A operação decreaseMax(h, val) irá:
  - Trocar o valor máximo max na raiz pelo valor val, SE val < max
  - SE o valor na raiz foi trocado, aplica-se o algoritmo maxHeapify a partir da raiz, para garantir que o heap continua em ordem Max-Heap

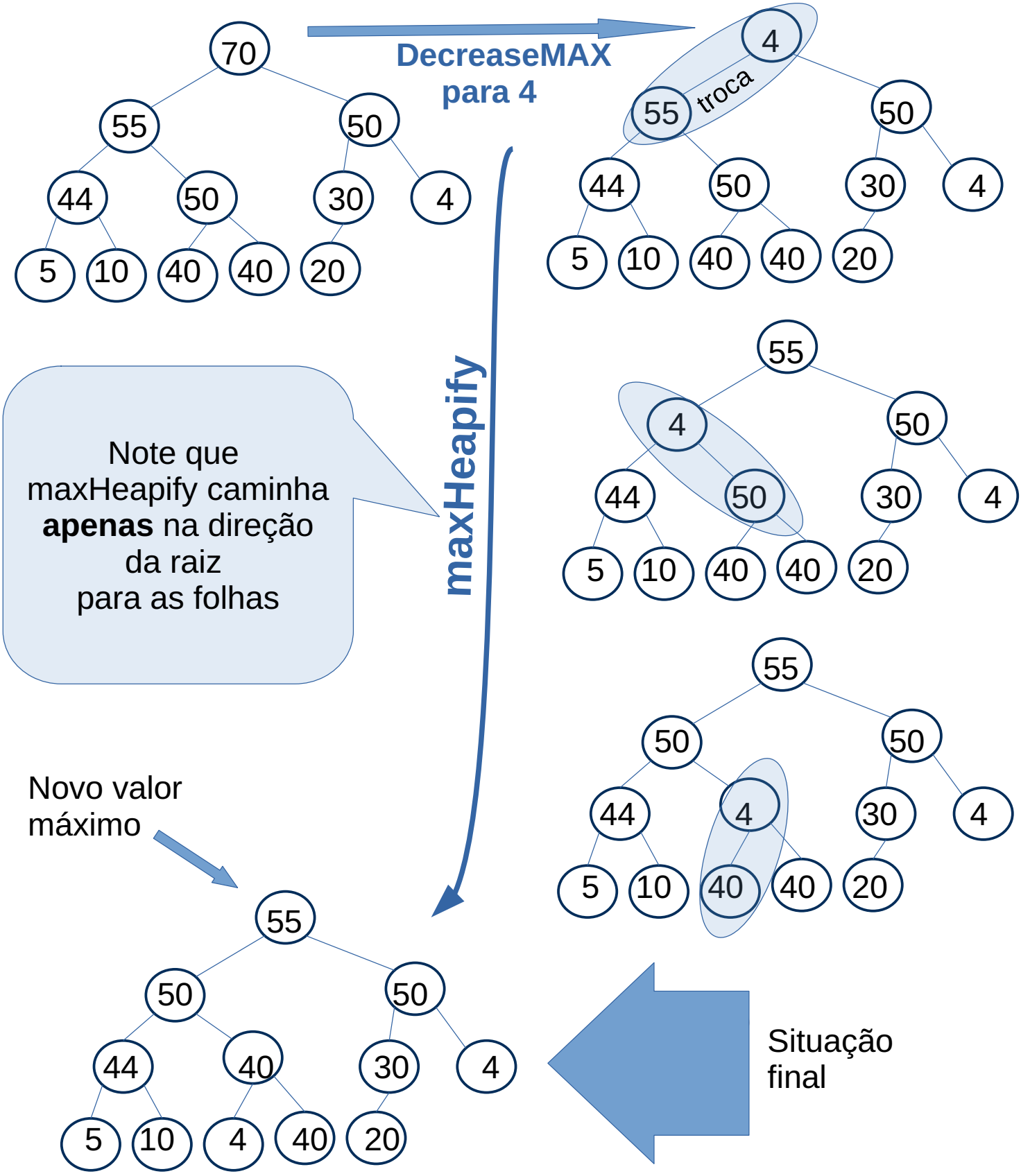
decreaseMax no Max-Heap

Dado o Max-Heap abaixo, usaremos decreaseMAX para 4



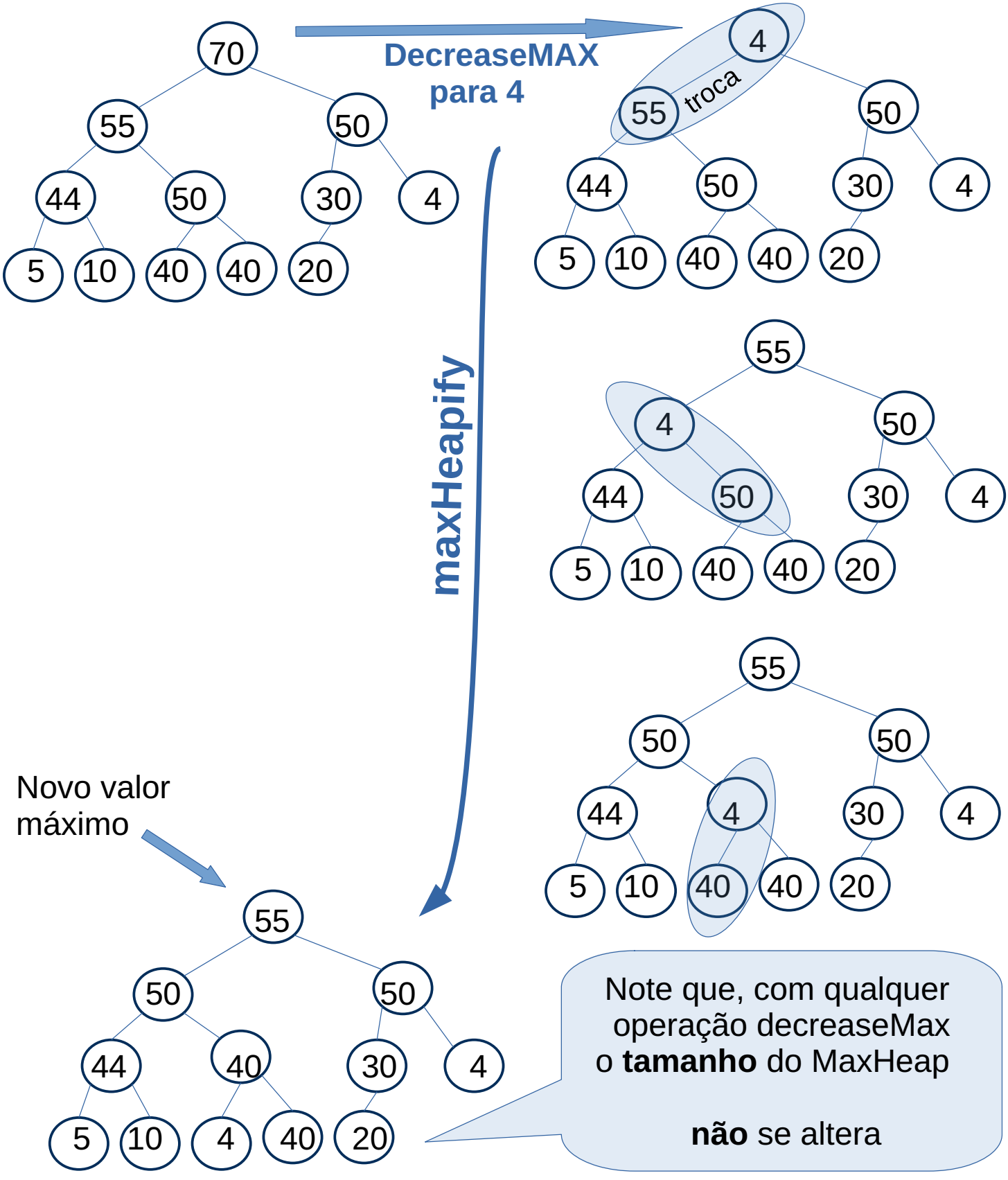
decreaseMax no Max-Heap

Dado o Max-Heap abaixo, usaremos decreaseMAX para 4



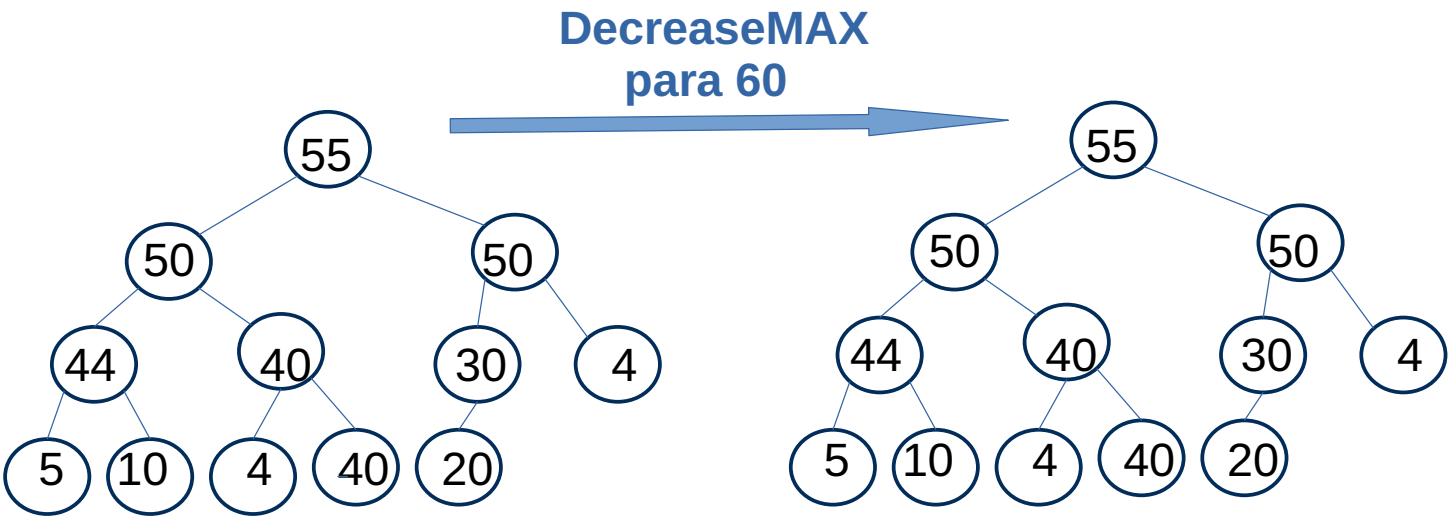
decreaseMax no Max-Heap

Dado o Max-Heap abaixo, usaremos decreaseMAX para 4



decreaseMax no Max-Heap

Suponha o Max-Heap conforme abaixo,
Vamos aplicar a operação decreaseMAX para 60



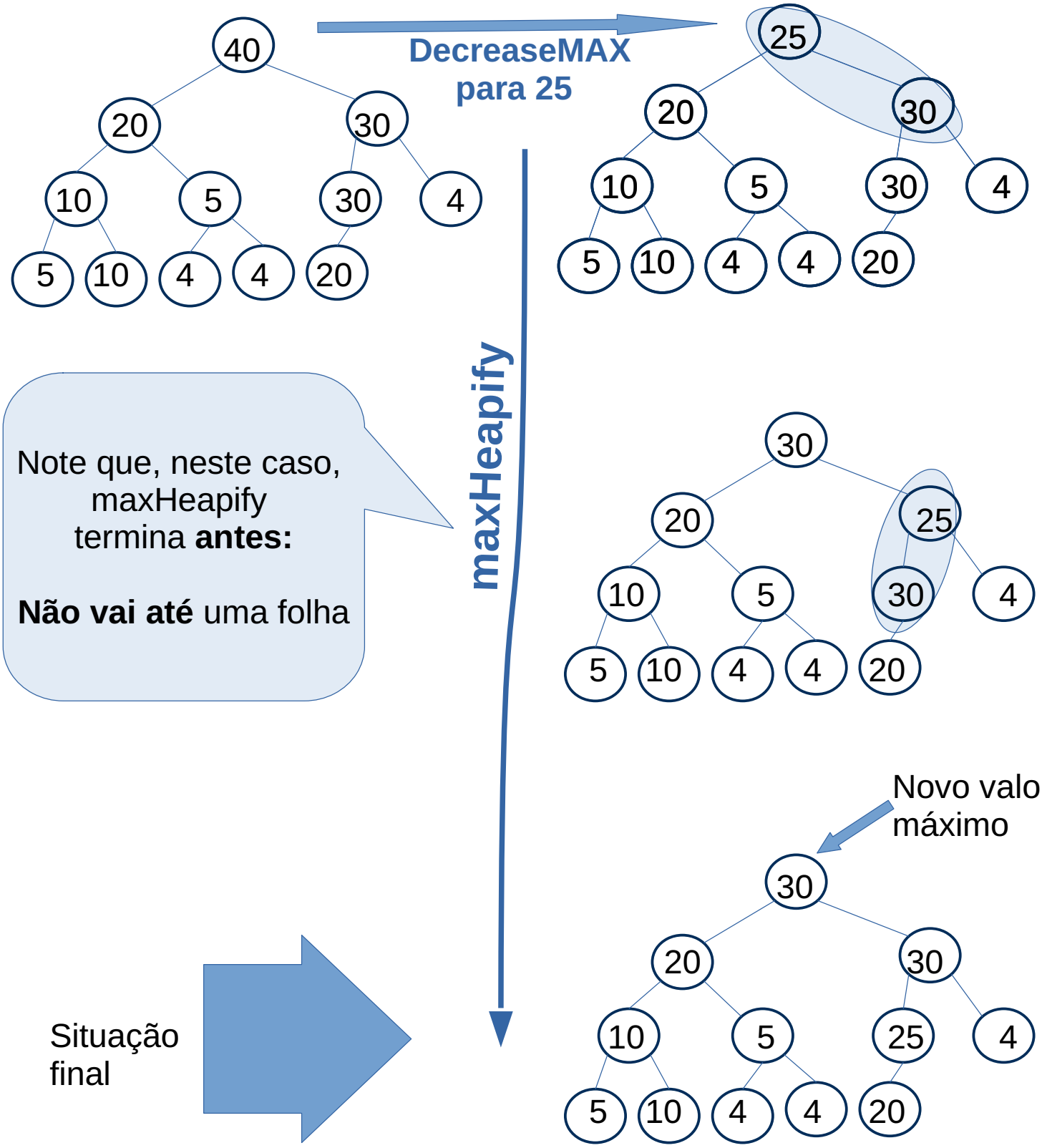
Note que
o Max-Heap
não sofre alterações

quando:

60 ≥ max

Mais um exemplo de decreaseMax no Max-Heap

Dado o Max-Heap inicial abaixo, aplicaremos decreaseMAX para 25



Aplicação do Max-Heap e operação decreaseMax

Podemos aplicar uma estrutura de dados do tipo Max-Heap e a operação decreaseMax no seguinte problema:

Dado um conjunto C de n valores, queremos obter o subconjunto S com os k menores elementos de C.

- Obs: supondo k <= n

Um algoritmo possível para resolver o problema acima seria:

- Extrair ***k*** elementos de ***C*** e criar um Max-Heap ***h*** com esses elementos.
  OBS: note que o Max-Heap tem tamanho k

- Para cada elemento ***e*** do conjunto C restante, aplicar a operação ***decreaseMax(h, e)*** no heap.

- Ao final, o Max-Heap ***h*** contém o subconjunto de k menores elementos de C

  - Vamos denominar esse algoritmo de ***acharKMenores***